(2)

NCS TIB 90-11

NATIONAL COMMUNICATIONS SYSTEM

AD-A230 399

TECHNICAL INFORMATION BULLETIN

90-11

MULTIPROTOCOL GATEWAY CAPABILITIES DEMONSTRATION REPORT

APRIL 9, 1990

OFFICE OF THE MANAGER
NATIONAL COMMUNICATIONS SYSTEM
WASHINGTON, D.C. 20305

April 9, 1990 Final

Multiprotocol Gateway Capabilities Demonstration Report

C-DCA100-87-C-0063

Booz-Allen & Hamilton, Inc.
4330 East West Highway
Bethesda, MD 20814

National Communications System
Office of Technology & Standards
Washington, DC 20305-2010

NCS TIB 90-11

Approved for Public Release; distribution is unlimited.

The National Communications System is working to promote standards within the data communication and telecommunication fields to increase the interoperability of diverse communication systems. These interoperability efforts improve the NS/EP communication capabilities of the nation by providing additional interconnectivity among Federal telecommunication assets. One such effort is determining the feasibility of using the excess transmission capacity of the Integrated Services Digital Network (ISDN) out-of-band signaling channels to reconstitute distrupted networks. Present telecommunication networks use one of many different protocol suites to interconnect their services. Networks employing different protocols cannot communicate with one another because the diverse protocols are incompatible. To address the protocol incompatibility issue, we recommend developing mechanisms, such as gateways that provide protocol conversions, to interconnect networks using diverse protocols.

ISDN
Networks

Data Communication
NS/EP

47

Unclassified Unclassified Unclassified

MULTIPROTOCOL GATEWAY CAPABILITIES DEMONSTRATION REPORT

Accession For		
NTIS GRA&I	X	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A-1		

APRIL 9, 1990

BOOZ•ALLEN & HAMILTON INC.
4330 EAST WEST HIGHWAY
BETHESDA, MD 20814

PREPARED FOR
OFFICE OF THE MANAGER
NATIONAL COMMUNICATIONS SYSTEM
8TH & SOUTH COURTHOUSE ROAD
ARLINGTON, VA 22204

UNDER CONTRACT NO. DCA100-87-C-0063

TABLE OF CONTENTS

LIST OF EXHIBITS

1.0 INTRODUCTION

The Office of the Manager, National Communications System (OMNCS), has been tasked to ensure the provision of National Security and Emergency Preparedness (NS/EP) telecommunications for the Federal Government under all conditions. In support of this task, the OMNCS is working to promote standards within the data communication and telecommunication fields to increase the interoperability of diverse communication systems. These interoperability efforts improve the NS/EP communication capabilities of the nation by providing additional interconnectivity among Federal telecommunication assets. One such effort is determining the feasibility of using the excess transmission capacity of the Integrated Services Digital Network (ISDN) out-of-band signaling channels to reconstitute disrupted networks.

1.1 BACKGROUND

Present telecommunication networks use one of many different protocol suites to interconnect their services. Networks employing different protocols cannot communicate with one another because the diverse protocols are incompatible. To address the protocol incompatibility issue, the OMNCS recommended developing mechanisms, such as gateways that provide protocol conversion, to interconnect networks using diverse protocols. These devices will act as translators between the two networks while maintaining communications with each network in its native language.

In the data communications field there are a number of prevalent standards for supporting end-to-end communications. Some of the existing sets of protocols that the OMNCS has identified to provide interoperability are:

- The International Telegraph and Telephone Consultative Committee's (CCITT) X.25 Recommendations
- The CCITT's recommendations for Integrated Services Digital Networks (ISDNs)
- The U.S. Department of Defense (DoD) developed Transport Control Protocol (TCP)/Internet Protocol (IP).

Since these standards are expected to coexist in the near future, gateways are needed that will allow users of these diverse protocols to communicate.

Earlier OMNCS efforts focused on specifying a gateway between X.25 and TCP/IP. Both a gateway development and description [1], and a specification and description language (SDL) document [2] were developed. These documents provide information to begin software development of an X.25-TCP/IP gateway that would allow users of the two protocols to exchange information.

More recent OMNCS efforts have focused on developing a network component to allow X.25 end user traffic to use future ISDNs as transport networks to interconnect disrupted portions of their packet-switched networks. This work investigated the use of the excess transmission capacity of ISDN out-of-band signaling channels to reconstitute disrupted networks. To assess this capability, the two protocols were compared, followed by a definition of protocol operating states and transitions among states needed to control and implement an X.25-ISDN gateway. Based on these preliminary operating states, a functional SDL document for an X.25-ISDN gateway was developed. The X.25-ISDN gateway has been implemented and its capabilities demonstrated. The X.25-ISDN gateway and demonstration are described in X.25-ISDN Gateway High Level Language Program Demonstration [12].

Building upon this previous work, an SDL description for a gateway designed to interconnect CCITT X.25 end users, Defense Data Network (DDN) X.25 (TCP/IP) end users, as well as CCITT X.25 and DDN X.25 (TCP/IP) end users, over the ISDN D channel was developed. The specification outlines the internal functionality of gateway components, their modules, and the interrelationship between these modules.

1.2 PURPOSE

The purpose of this document is to:

- Provide a description of the multiprotocol gateway and its associated hardware and software

- Describe the operational gateway demonstration that took place at Booz, Allen & Hamilton on February 28, 1990

- Provide results and conclusions drawn from the demonstration.

1.3 PROJECT REFERENCES

The following documents have been used in the development of this document:

1. X.25/PLP/TCP Gateway Development and Description. National Communications System. February 1986.

2. X.25/PLP-DDN/TCP Gateway SDL Technical Specification. National Communications System. June 1986.

3. X.25 and ISDN Packet Mode State Transition Tables. National Communications System. November 1987.

4. X.25-ISDN Gateway State Transition Tables. National Communications System. May 1988.

5. Data Communications Networks: Interfaces. Recommendations X.20-X.32. CCITT, Vol. VIII - Fascicle VIII.3. 1984.

6. Integrated Services Digital Network (ISDN). Series I Recommendations. CCITT, Vol. III - Fascicle III.5. 1984.

7. X.25-ISDN Gateway SDL Description. National Communications System. August 1988.

8. Functional Specification and Description Language (SDL) Recommendations Z.101 - 104. CCITT, Vol. VI - Fascicle VI.10. 1984.

9. Defense Data Network X.25 Host Interface Specification. Defense Communications Agency. December 1983.

10. Internet Protocol. MIL-STD-1777, Defense Communications Agency, J110. 12 August 1983.

11. Transmission Control Protocol. MIL-STD-1778, Defense Communications Agency. 12 August 1983.

12. X.25-ISDN Gateway High Level Language Program Demonstration. National Communications System. May 12, 1989.

13. Multiprotocol Gateway SDL Description. National Communications System. August 23, 1989.

The foundation of the multiprotocol gateway is composed of the CCITT protocol specifications (5 and 6) and the DDN protocol specifications (9 though 11). Other references include documents relating to earlier efforts in support of a X.25-TCP/IP gateway (1 and 2) and efforts involving the X.25-ISDN gateway (3 and 4). The methodology and syntax used to specify the gateway is based on a separate series of CCITT recommendations -- Z series (8).

1.4 ORGANIZATION

The Multiprotocol Gateway Capabilities Demonstration Report is composed of five sections. Section 2 focuses on the gateway architecture. It describes the environment in which the gateway will operate and includes a description of the gateway components, specific hardware, and gateway interfaces. The gateway software is described in Section 3. Section 4 contains a description of the gateway demonstration. Results and conclusions drawn from the demonstration are contained in Section 5.

2.0 GATEWAY ARCHITECTURE

The multiprotocol gateway is designed to support the interconnection of CCITT X.25 end users, DDN X.25 (TCP/IP) end users, as well as CCITT X.25 and DDN X.25 (TCP/IP) end users, over the ISDN D channel. Consistent with the mission of the OMNCS, the gateway is designed to support emergency communications. This requires that the gateway be transportable and use hardware that is readily available. For these reasons, the gateway has been developed for a 386-based microcomputer using the DOS operating system. However, the modular design approach to the gateway is applicable to the design of larger gateways, such as a Digital Equipment Corporation MicroVAX II based gateway that could support multiple connections at a faster throughput rate.

2.1 GATEWAY ENVIRONMENT AND COMPONENTS

The environment in which the multiprotocol gateway will be operating is shown in Exhibit 2-1. As pictured, the environment is divided into four sections: the X.25 network, the DDN (TCP/IP) network, the ISDN, and the gateway. For the gateway to be useful it is necessary that the network users operate in their native mode. It is the gateway's responsibility to make up for all incompatibilities. Network users are not required to have any additional equipment or make any modifications to their configuration when communicating across networks.

The gateway can be subdivided into the following four components:

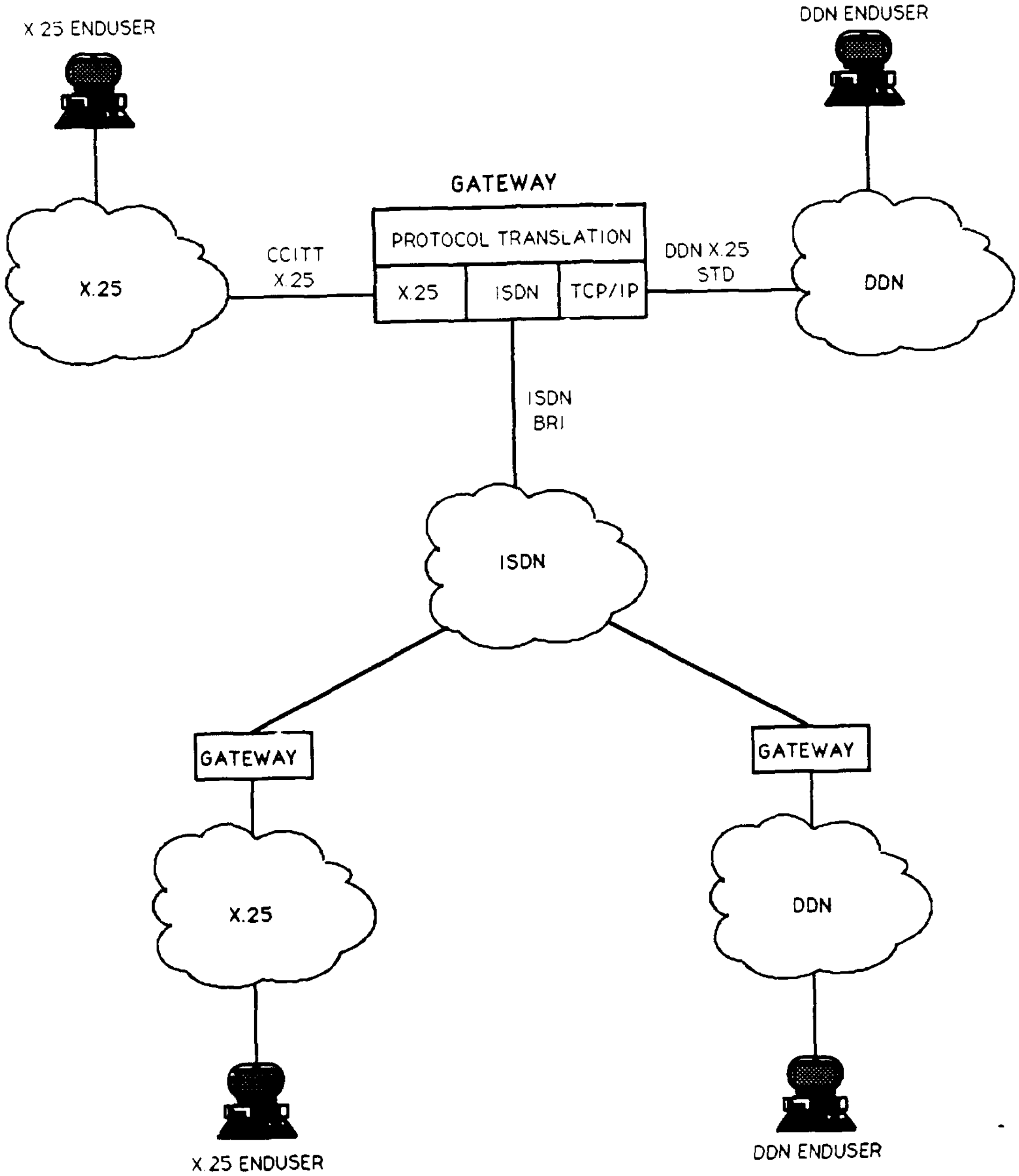
- CCITT X.25 network interface
- DDN X.25 (TCP/IP) network interface
- ISDN Terminal Adaptor (TA) interface
- Protocol translation.

The CCITT X.25 network interface can be configured to appear as a standard data terminal equipment (DTE) device connected to an X.25 network or it can be configured to appear as a standard data circuit-terminating equipment (DCE) device supporting an X.25 end user. The CCITT X.25 network interface will perform all addressing, flow control, and call setup/clearing that are normally part of the X.25 protocol. The CCITT X.25 hardware and software are off-the-shelf, commercially-developed products.

Similarly, the DDN X.25 (TCP/IP) network interface can be configured to appear as a standard DTE device connected to the DDN or it can be configured to appear as a standard DCE device supporting a DDN X.25 end user. The DDN X.25 network interface will perform all addressing, flow control, and call setup/clearing that are normally part of the TCP/IP protocol.

EXHIBIT 2-1

Multiprotocol Gateway External Environment



The DDN X.25 (TCP/IP) hardware and software are off-the-shelf, commercially-developed products.

The ISDN TA interface provides the basic rate interface (BRI) to the ISDN. The ISDN BRI supports two 64 kilobits per second (kbps) circuit-switch channels (referred to as B channels) and one 16 kbps packet data channel (D channel). The D channel serves two main purposes. First, it carries common-channel signaling information to control circuit-switched calls on associated B channels at the user interface. In addition, the D channel may be used for packet-switching (9600 bps) or low-speed (100 bps) telemetry at times when no signaling information is waiting. The multiprotocol gateway uses the excess bandwidth in the D channel to provide packet-switched transport services. The ISDN TA hardware and software are off-the-shelf, commercially-developed products.

The protocol translator will be the interface between the three end user types. It will perform packet translations, maintain address tables, track status of the connections, and provide the man-machine interface with the gateway. The protocol translator has been developed and implemented using the Microsoft "C" programming language.

2.2 HARDWARE

The multiprotocol gateway is composed of two types of components; a master processing unit (MPU) and a network interface card (NIC). Each component is designed around an independent microprocessor unit, supporting memory, and input/output controllers.

Each operational gateway configuration is composed of a single MPU controlling gateway operation and three NICs to provide the interface for each of the physical connections supported. Exhibit 2-2 shows the four basic gateway components, their interrelationships, and associated external interfaces.

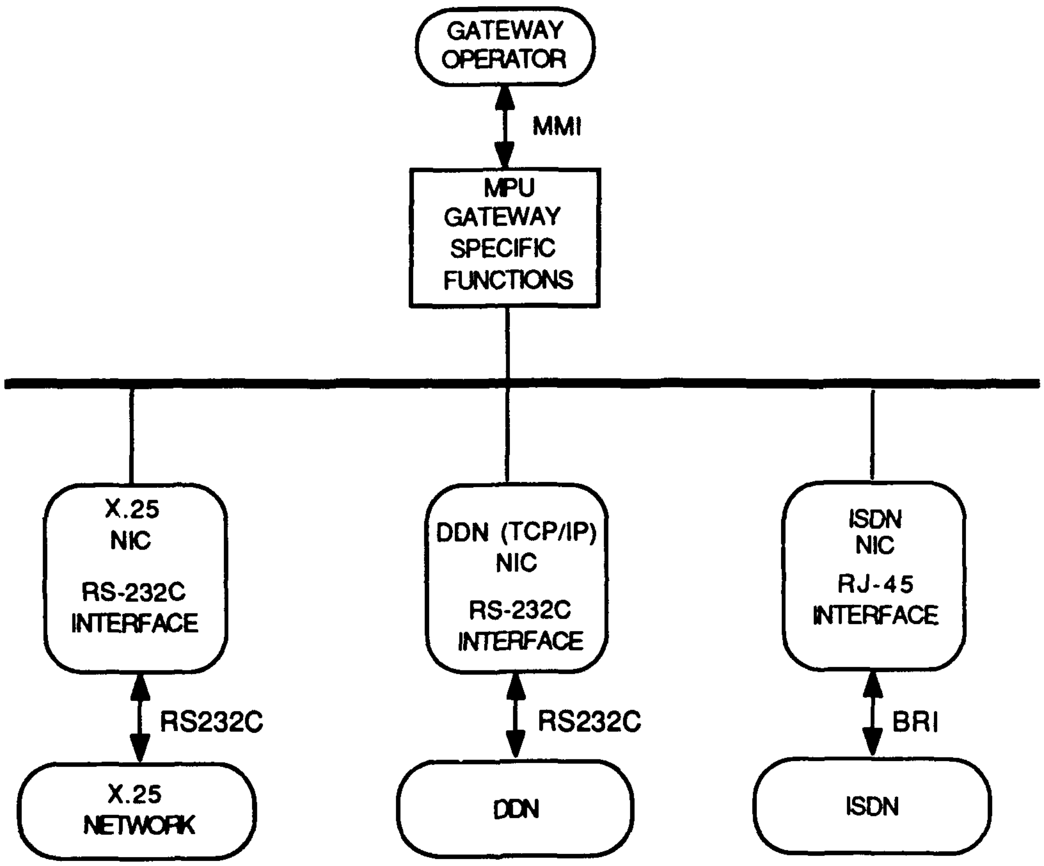
2.2.1 Master Processing Unit

The MPU controls all gateway operations, including:

- User interface
- Protocol and packet translations
- Address mappings
- Packet routing
- Statistics gathering and error reporting
- Packet buffering
- Real-time scheduling of multiple independent processes
- Communications across components.

EXHIBIT 2-2

Multiprotocol Gateway Components



The MPU selected for the multiprotocol gateway is based on a Compaq 386 super microcomputer with the following characteristics:

- 32 bit microprocessor; based on an INTEL 386 chip
- 4 megabyte (MB) main memory
- 20 megahertz (MHz) internal clock
- Adjustable cache memory
- 120 MB direct access storage
- 1.2 MB floppy disk.

This configuration will support system development and testing as well as the real-time operation of the gateway.

2.2.2 Network Interface Cards

Three types of NICs are to be used in the multiprotocol gateway. The first NIC provides the interface to the CCITT X.25 network, the second supports the interface to the DDN X.25 (TCP/IP) network, and the third provides the terminal adaptor capabilities required for the ISDN. The NICs perform the unique protocol processing functions required of the physical, link, and network layers of the specific networks being connected. The DDN X.25 NIC also supports the transport layer functions provided by TCP. All of the NICs are personal computer adaptor boards and interface directly to the MPU's input/output bus.

2.2.2.1 CCITT X.25 NIC. The EICON X.25 PC card has been selected for the CCITT X.25 NIC. The card has the following characteristics:

- 32 bit microprocessor; based on Motorola 68008 chip
- 512 kilobytes (kB) random access memory (RAM)
- 3.68 MHz internal clock
- 1 Z8530 serial communications controller
- 4 direct memory access channels.

All processing is controlled through software downloaded from the MPU, requiring no off-line program storage. Configuration parameters used for the EICON X.25 interface card in the multiprotocol gateway implementation are as follows:

- Memory Port Addressing - CA00
- I/O Port Addressing - 380
- PC Bus Interrupt - IRQ3
- Software Interrupt Vector - 5C

Refer to the EICON Technology Adapter card Installation Guide for more information.

2.2.2.2 DDN X.25 (TCP/IP) NIC. The DDN X.25 (TCP/IP) NIC selected is the Frontier Technologies Corporation's AdCom2-I Intelligent Communication Controller. The card has the following characteristics:

- 80188 microprocessor
- 512 kB RAM
- Z80 based communications controller
- 4 direct memory access channels.

All processing is controlled through software downloaded from the MPU, requiring no off-line program storage. Configuration parameters used for the AdCom2-I interface card in the multiprotocol gateway implementation are as follows:

- Memory Port Addressing - D000
- I/O Port Addressing - 340
- PC Bus Interrupt - IRQ11
- Software Interrupt Vector - 6A

Jumper settings for the AdCom2-I are as follows:

- Reset (JP1) - none
- ROM Size (JP2, JP3, JP4) - all set at "AB"
- RT (JP5) - XT/AT position, "BC"
- High Order Address (JP6) - At/Rt position, "AB"
- RAM Size (JP7) - 256k position, "AB"
- Clock Rate (JP8) - "BC" position.

Refer to the AdCom2-I Plus User/Programming Manual for more information.

2.2.2.3 ISDN NIC. The ISDN NIC selected is the TELEOS B100PC Communications Coprocessor. The B100PC Communications Coprocessor is an IBM PC/XT/AT-compatible basic rate (2B+D) adapter, providing ISDN S/T interface compatibility in accordance with CCITT recommendations at the physical, data link, and network layers. The card has the following characteristics:

- 68HC000 processor
- 512 kB RAM
- 12.288 MHz clock
- 4 direct memory access channels.

Similar to the X.25 NIC, all processing is controlled with real-time software that is downloaded from the MPU and requires no off-line storage. The TELEOS equipment has been used previously in the development and implementation of the X.25-ISDN Gateway [12]. Configuration parameters used for the Teleos terminal adapter card in the multiprotocol gateway implementation are as follows:

- Memory Port Addressing - B000
- I/O Port Addressing - 100
- PC Bus Interrupt - IRQ5
- Software Interrupt Vector - 60

Refer to the Teleos B100PC Communications Coprocessor Users Manual for more information.

2.3 INTERFACES

The multiprotocol gateway provides four external interfaces: a gateway/CCITT X.25 interface, a gateway/DDN X.25 interface, a gateway/ISDN interface, and a man-machine interface (MMI). These interfaces are shown in Exhibit 2-2.

2.3.1 Gateway/CCITT X.25 Network Interface

The gateway/CCITT X.25 interface is provided through the EICON NIC. The EICON NIC supports the following:

- RS-232-C network interface
- 32 virtual circuits
- 9 concurrent host sessions
- Selectable, internal/external clocking
- Data Rates Supported: 2400, 4800, 9600, 19200, 38400, 48000, 56000, 57600, 64000.

2.3.2 Gateway/DDN X.25 Interface

The gateway/DDN X.25 interface is provided through the AdCom2-I NIC. The AdCom2-I NIC supports the following:

- RS-232-C network interface
- 32 virtual circuits
- Selectable, internal/external clocking
- Data Rates: 2400, 4800, 9600, 19200, 38400, 48000, 56000.

2.3.3 Gateway/ISDN Interface

The gateway/ISDN interface is provided through the Teleos B100PC Communications Coprocessor NIC. The B100PC supports the following:

- Standard 8-pin ISDN RJ-45 interface
- Standard 6-pin analog phone RJ-11 interface
- ISDN BRI (2B+D); Line Rate, 192 kbps (nominal)
- B channel data rate, 64kbps
- D channel data rate, 16kbps
- D channel X.25 packet data rate, 9.6kbps.

2.3.4 Man-Machine Interface (MMI)

The MMI provides a menu-driven interface so the gateway operator can perform the following gateway functions:

- Gateway start-up
- Statistics gathering and reporting
- Gateway shutdown
- Address and routing table modifications.

The interface is provided through a standard red, blue, green (RGB) color monitor and IBM-AT compatible keyboard.

2.4 MINIMUM EQUIPMENT CONFIGURATION

The multiprotocol gateway should be implemented on a 286 or 386 based AT machine running DOS with the following components installed:

- 1 MB of onboard memory
- 1.2 MB floppy disk drive or hard disk
- 1 EICON X.25 PC card
- 1 Frontier Technologies Corporation AdCom2-I Intelligent Communications Controller
- 1 TELEOS B100PC Communications Coprocessor.

3.0 GATEWAY SOFTWARE

The purpose of the gateway software is to provide the protocol translation capabilities of the TCP-IP/X.25/ISDN Gateway system. This translation ability is extensive enough to allow full connectivity through the gateway (i.e., any user type is allowed to connect to any other user type).

The program is designed to be portable and runs in the DOS operating system. The program was implemented in the "C" programming language using the Microsoft Quick C compiler. The platform used to support the gateway system can be any 286 or 386 based machine with standard RGB monitor. The program design allows easy tailoring of major operational parameters. The current settings of these parameters, as well as the method by which they can be changed are contained in Exhibit 3-1.

This program provides for transparent operation within the context of the end user networks. The program design and implementation provide for minimum effect to end user nodes of the existing network. For limitations of this transparency in the current implementation see Section 3.3.

The operating environment of the gateway is illustrated in Exhibit 3-2. As mentioned above, each of the network interfaces is fully connected to allow communication along all of the possible paths. The man-machine interface is implemented in menu format to provide for ease of use and flexibility of operation. The gateway operator is assumed to be a passive operator. Once the gateway has been initialized no further operator intervention is required.

3.1 OPERATIONAL DESCRIPTION

To support the protocol translation function the gateway has the ability to accept incoming calls and data, determine the next required connection in the desired end user circuit, and request the call or transmit the data. All routing decisions are made at time of call initiation. These decisions are based on operator entered address translation and gateway location values. Gateway locations are represented by the ISDN address of the gateway. Exhibit 3-3 illustrates the decision flow chart used to determine the next link routing.

For gateway to gateway links the source gateway determines the ultimate destination protocol type and transmits it along with the destination and source address information to the remote gateway. A special call header packet is transmitted immediately after the ISDN call has been established. Upon receipt of the

Exhibit 3-1

Gateway Parameter Settings

Operation	Current Value	Method of Change
# of supported end users	50	Change constant MAX_TBL in "gatedef.h" (1,2)*
Max # of concurrent open ISDN links	4	Change MAX_ISDN_OUT constant in "gatedef.h" (1,2,3)*
Max # of concurrent X.25 links	4	Change MAX_X25_OUT constant in "gatedef.h" (1,2)*
Max # of concurrent open TCP links	4	Change MAX_TCP_OUT constant in "gatedef.h" (1,2,4)*
X.25-TCP Inter-address mapping entries	N/A	Create/Modify translation table
ISDN location	N/A	Create/Modify translation table entries
ISDN gateway address	0002	Modify variable "loc_isdn_addr" in "gatedev.c" (2)*
TCP gateway address	010.000.000.052	Modify "NET.TCP"
X.25 board interface parameters	refer to paragraph 2.2.2.1	Modify X25START.BAT
TCP board interface parameters	refer to paragraph 2.2.2.2	Modify TCP_START.BAT
TCP network interface parameters	refer to paragraph 2.2.2.2	Modify NET.TCP
DDN/X.25 network interface parameters	refer to paragraph 2.2.2.2	Modify NET.X25
ISDN board interface parameters	refer to paragraph 2.2.2.3	Modify ISDNSTART.BAT

* see notes

Exhibit 3-1

Gateway Parameter Settings
(cont.)

Notes:

1. Increases to this value will affect speed and memory used

2. These change require recompile before taking affect

3. The interface board requires prior notification of the number of listening circuits (incoming calls from other gateways). This number is currently set at 4 and will not be affected by changes to the above constant.

4. Refer to paragraph 3.3.

Exhibit 3-2

Gateway Operational Configuration

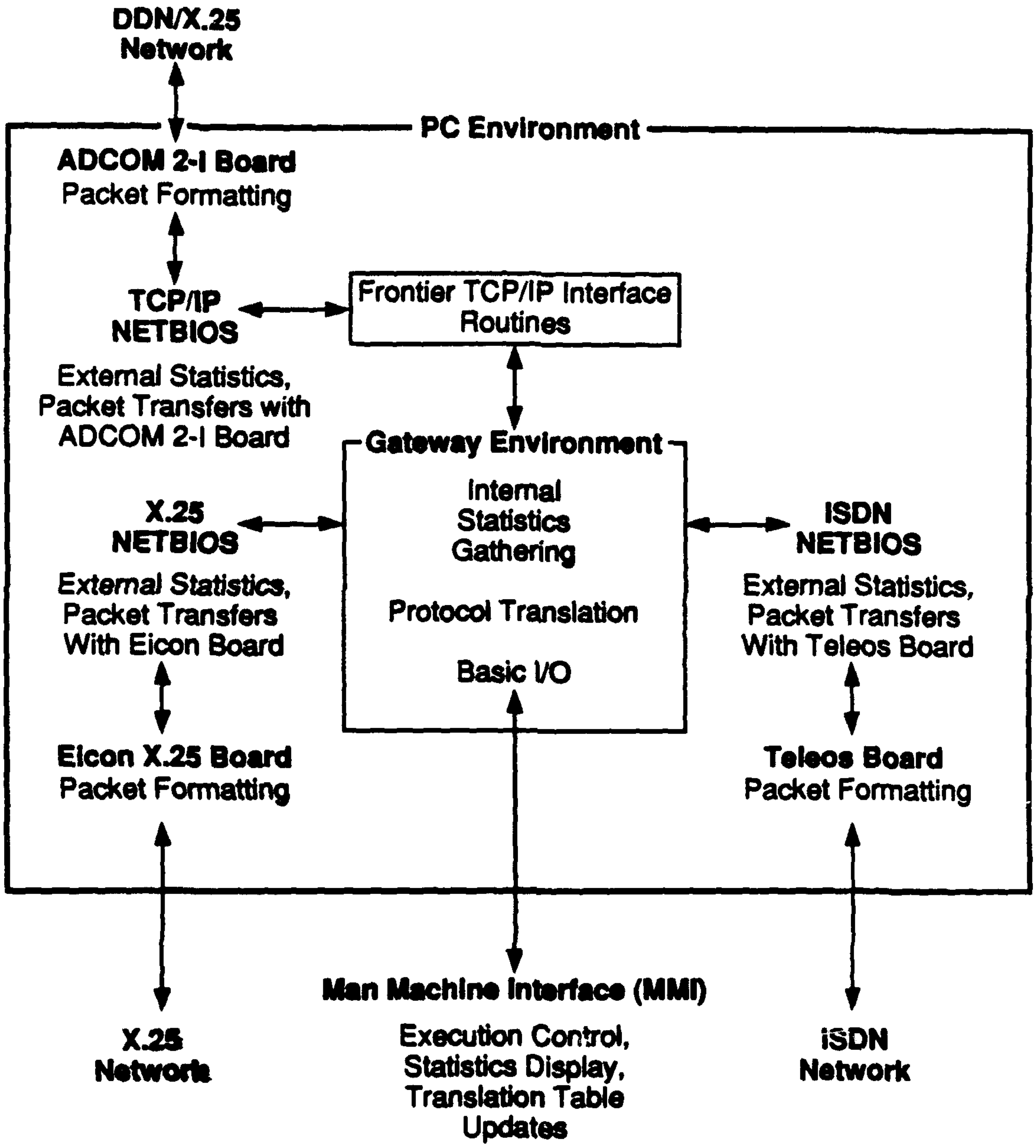
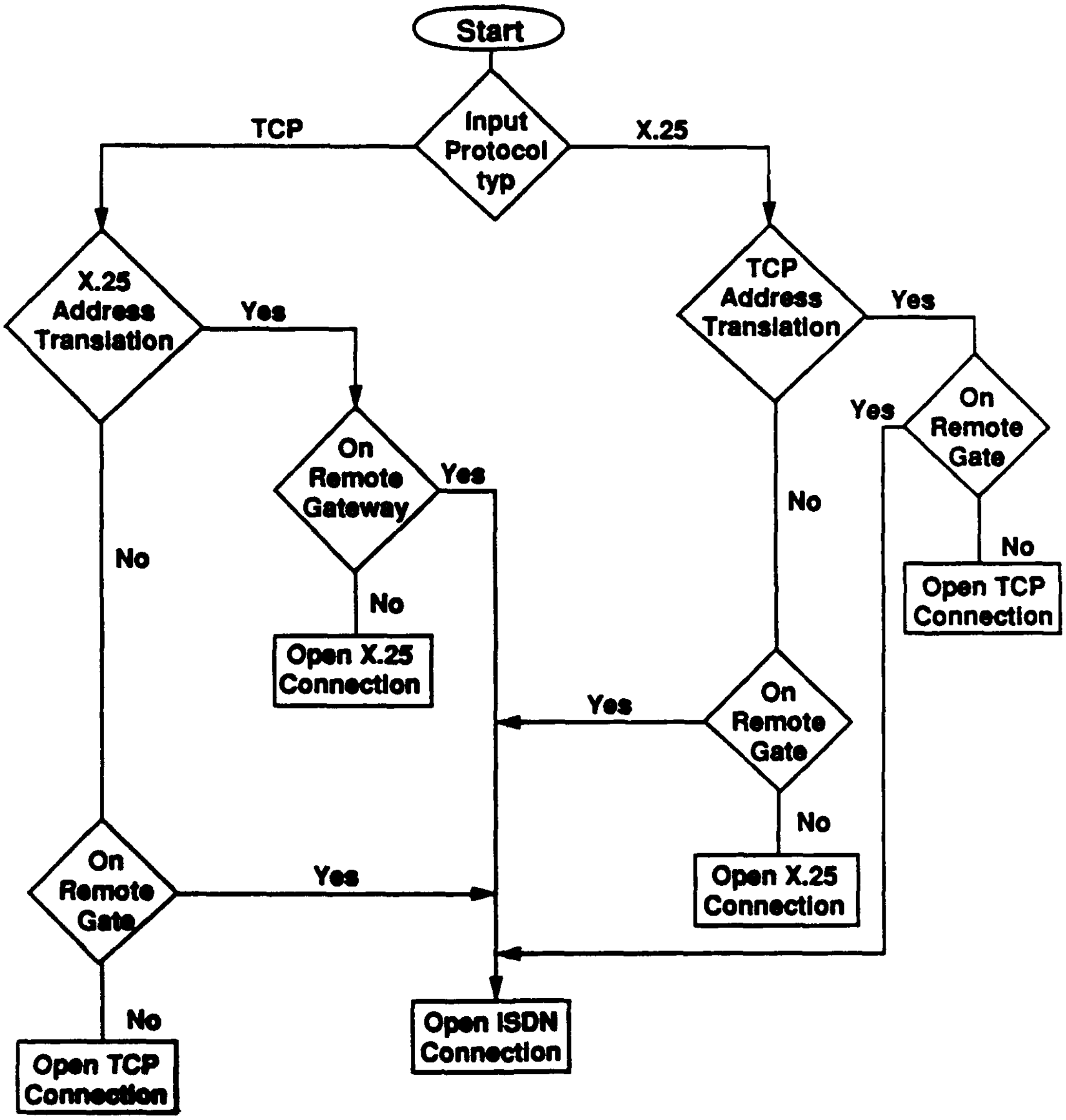


Exhibit 3-3

Gateway Routing Decision Chart



call header the remote gateway sets up the next link in the connection based on the information contained. The call header information is not transmitted on the destination link. Errors in any of the above network actions will result in immediate cancellation of the call from the source node.

Adapter board interfaces are handled through specified interrupt vectors. These interrupt vectors must be supplied prior to system initialization (see Exhibit 3-1). The values of the card setup parameters will vary from platform to platform depending upon configuration. Direct adapter hardware interfaces are handled by proprietary software modules and initiated by the generation of the DOS interrupt specified as part of the setup file. For the purpose of the following discussion these modules will be referred to by the general term NETBIOS.

The gateway software provides a man-machine interface to allow the gateway user to configure the gateway to support the current operational environment, initialize it, monitor throughput, and gracefully halt its execution. The gateway operator is assumed to be a passive operator and is therefore not provided with any command that will initiate, change, or otherwise affect the flow of data packets through the gateway when it is operating. Due to the limited nature of system level resources extensive operator use of commands such as the statistics command will slow the processing of data packets.

Finally the gateway software provides information gathering capabilities on overall gateway operations. Two types of information are provided: trace information and statistical information. Trace information traces the flow of messages through the gateway. Statistical information gives a breakdown of the messages processed overall, relative to a single protocol or relative to a single open link.

3.2 FUNCTIONAL BREAKDOWN

The gateway system is composed of five modules. These modules perform the logical functions required to translate from one protocol to another as well as the other support functions listed in Section 3.1. These modules are:

- Man-Machine Interface - Supports the user interface
- Gateway State Handling - Manages start-up and shutdown of the gateway
- Gateway Driver - Schedules translation oriented activities as required

- Call Processing - Processes received call requests, routing them to their destination

- Data Transfer Processing - Performs the physical data transfer once a call has been established.

The following subsections discuss each of the above modules in detail as well as their further decomposition into tasks and routines.

3.2.1 Man-Machine Interface

The Man-Machine Interface module manages the interface between the gateway operator and the gateway functions. There are three types of interfaces with the operator: command flow from the operator, information flow from the operator and information flow from the gateway. Each of these interfaces corresponds to an individual task within the MMI module. These tasks are discussed in the following subsections.

3.2.1.1 Command Handling Task. This task is responsible for the processing of operator commands. It displays the necessary menus, waits for operator input and transfers control to the appropriate task or module depending on the command entered. This module is also responsible for providing sufficient prompting to the operator to allow for ease of use.

Upon program start this task displays the command entry menu and prompts the operator for a command to perform. This task will then cycle continuously checking for either operator input or network input to process. This task calls the Gateway Driver module to schedule and/or perform network actions. Receipt of an operator command causes this module to verify the validity of the command. If the command is invalid this task indicates the error to the operator and waits for further input. If the command is valid this task will initiate the appropriate module/task as follows:

- If the command is a request for interface card initialization, gateway startup or call clearing, this task will pass control to the Gateway State module to change the gateway's readiness state

- If the command is for return to DOS this task will pass control to the Editing task to save the translation table and then exit to DOS when this is complete

- If the command is for statistics display this task will pass control to the Statistics task for display of the requested statistical breakdown

- If the command is for translation table editing this task will transfer control to the Editing task of the MMI module

- If the command is for trace activation/deactivation this task will record the requested state of the trace function.

The Command Processing task is composed of the following routines:

- main - responsible for accepting and processing commands

- main_menu - responsible for the display of the main menu.

3.2.1.2 <u>Statistics Handling Task</u>. The purpose of this task is to process requests for statistical display. This task displays gateway statistics grouped by overall protocol type, relative to a single protocol, and relative to a single active link. This task is responsible for displaying the appropriate screen and handling any interface required with the protocol NETBIOS to gather the link level statistics.

Upon receipt of a request for overall transmission statistics this task will display the previously captured protocol level statistics. Once the display is completed the task waits until the user has indicated that the next screen can be displayed. After a prompt has been displayed this task will cycle checking for both operator and network input processing any network input encountered. The Gateway Driver module is used to check for network activity. Upon receipt of operator input this task will display a screen containing at least the first 88 bytes of the last message received on each of the three protocol links. These messages are displayed in hexadecimal format. The task then enters another wait state. Receipt of operator indication to continue will then cause this task to return control to the Command Processing task.

Upon receipt of a request for a specified protocol transmission statistics the task will display an initial screen showing internal statistics collected about traffic to and from the protocol channel. This task will then enter a wait state to allow the operator to read the screen. Wait state processing is as discussed above.

Receipt of an operator indication that the task can continue causes the task to check for any open calls for the protocol. If an outstanding call is found this task will transmit a statistics request to the NETBIOS for information about the open link. Information returned from this request is formatted, displayed to the operator and a wait state is entered. This task will cycle through all open calls in this manner and then return control to the Command Processing task.

Where there are no open calls on the TCP or ISDN channels, control will return to the Command Processing module after display of the initial screen.

The Statistics task is composed of the following routines:

- Trans_stats - displays transmission statistics and last packet received
- Stat_scr - displays the standard statistics screen
- Disp_brk_stats - displays the common portion of the statistical breakdown
- X25_stats - displays X.25 related statistics
- TCP_stats - displays TCP related statistics and TCP link statistics
- ISDN_stats - displays ISDN related statistics and ISDN link statistics.

3.2.1.3 <u>Edit Handling Task</u>. The Edit Handling task oversees operator initiated translation table edits. This task is also responsible for the initial load and final save of any existing translation table entries.

Upon receipt of control at the time of system start this task will determine if there is a previously saved translation table. If a saved table is found then this task reads in the saved table and indicates that at least one entry exists. If a table file does not exist then this task indicates that the translation table is empty.

Upon receipt of an operator request to perform a table edit this task will display the first translation table entry to the operator along with the edit session menu. It will then wait for operator input. Continuous checks for network activity to process are performed during this wait period. Operator input is interpreted as follows:

- Request to edit current entry - opens current entry for edit and displays the entry edit menu
- Request to edit specified address - requests address from operator, finds address in translation table, opens entry for edit, displays the entry edit menu

- Scroll to next entry - displays values associated with next table entry

- Create entry - finds an empty translation table entry, opens it for edit, blanks input fields and displays entry edit menu

- Exit - returns control to the Command Processing task.

Upon notification that the gateway is going off-line the task will save the current translation table, destroying any previously existing copies.

The Edit Handling task is composed of the following routines:

- Read_trans_tbl - reads the previously stored translation table

- Ed_trans_tbl - oversees the table edit session

- Display_edit_screen - displays all or part of the edit screen using the data and menu indicated by the caller

- Det_ed_act - determines next action to perform based on operator input

- Ed_sel_row - edits a selected row in the translation table

- Up_X25_addr,Up_TCP_addr,Up_ISDN_addr - accepts updates to the addresses, performing the necessary validation

- Get_input_addr - performs generalized actions required to input an address

- Store_trans_tbl - stores the translation table in a disk file for later use.

3.2.1.4 <u>Common Routines</u>. The MMI module maintains the following common routines which are used by all of its tasks:

- Output_message - outputs a message to the message area of the screen and to the log file if trace is enabled

- Wait_user_input - waits for user keyboard input while calling the Gateway Driver module to check for network activity to process.

3.2.2 Gateway State Handling

This module performs the actions required to initialize the gateway to an operating state and to gracefully return it to a non-operational state. The initialization processes include initialization of the interface cards and setting up for call processing. Gateway shutdown includes clearing all existing calls. The following subsections discuss these tasks in greater detail.

3.2.2.1 I/O Card Initialization Task. This task manages the initialization of the I/O adapter cards and their corresponding NETBIOS interface. Initialization processing is handled by manufacturer supplied initialization programs executed as batch files through the "C"/DOS interface. The following batch files are maintained to initialize the I/O adapter cards:

- ISDN_START.BAT
- TCP_START.BAT
- X25_START.BAT.

Initialization of NICs can be performed in trace or no-trace modes. Card initialization in trace mode will result in display to the operator of the intermediate steps in the process as well success or failure indications. The following batch files are maintained:

- ISDN_TRACE.BAT
- TCP_TRACE.BAT
- X25_TRACE.BAT.

The operator can choose to initialize only a subset of the supported links if all of the interface types are not available in the current environment. Only those protocols which have been initialized will be set up for call processing in the next task. There are no "C" routines in the I/O Initialization task.

3.2.2.2 Setup for Call Processing Task. This task sets up the previously initialized cards to receive their first incoming call. To set up an individual protocol's NETBIOS there must be at least one translation table entry and the card must have been previously initialized. Attempts to access cards that have not been set up for call processing will result in termination of the offending call. This task will indicate the presence of any uninitialized cards to the operator. Method of call setup varies by protocol type and is discussed in the following paragraphs.

X.25 Call Processing Setup

If the X.25 card has been initialized then this task will start a listen for each X.25 address contained in the translation table which can accept incoming calls. Incoming calls are possible if the address is located on a remote gateway or if it has a TCP translation. Gateway location is determined by comparing the end user's ISDN location to the local gateway address. Use of the facility matching capability of X.25 is ignored in the present implementation. Memory is allocated from dynamic memory to hold the listen requests. The pointer to the request is stored in the translation table entry for this address. Listen commands are initiated by sending a listen command to the X.25 NETBIOS.

TCP/IP Call Processing Setup

If the TCP/IP card has been initialized then this task will start a listen for each address/port ID combination contained in the translation table which is deemed capable of receiving calls. An address is deemed capable of receiving calls if it is on a remote gateway or it has an X.25 translation. The initiated listen will match only those calls destined to this address/port combination. Memory is allocated for each listen packet and a pointer to the packet is stored in the translation table entry. The translation table entry number is used as a user supplied process id in the packet for ease of reference later. Listens are initiated by sending the listen command to the TCP/IP NETBIOS.

ISDN Call Processing Setup

If the ISDN card and corresponding NETBIOS have been initialized this task will perform the necessary actions to allow the gateway to receive ISDN calls. Calls are made to the ISDN NETBIOS SIP and X25D DPP to establish logical sessions for the current gateway session. Using the session numbers the X25D processes are initialized to contain the current gateway operational parameters for this link. Finally, a Receive is queued with the X25D driver to receive the next driver to application notification regarding the current session.

Setup for Call Routines

The Setup for Call Processing task is composed of the following routines:

- Setup_gate - oversees the gateway setup function
- Init_X25_listen - determines which X.25 addresses require listens
- X25_listen - queues a listen with the X.25 NETBIOS for the

given address

- Init_TCP_listen - determines which TCP addresses require listens

- TCP_listen - queues a listen with the TCP NETBIOS for the given address/port combination

- ISDN_X25D - requests logical session number from the ISDN X25D process of the NETBIOS

- ISDN_SIP - requests the logical session number from the ISDN SIP process of the NETBIOS

- ISDN_start_DPP - loads the X25D process with the current session information

- ISDN_call_wait - queues a receive request with the DPP to receive the next session information packet.

3.2.2.3 Call Clearing Task. Upon receipt of an operator command to clear the existing calls this task will perform the necessary actions to shut down the I/O adapter card and corresponding NETBIOS processing. X.25 hangups and command cancels are issued for each of the currently open X.25 links. The task will hang up the ISDN logical session that cancels all outstanding actions and calls in that session. For TCP/IP links this task will issue abort commands to the TCP NETBIOS, closing the links and cancelling pending receives.

Upon completion of the above processing this task will loop through the translation table de-allocating previously allocated listen packets. Once this process is complete this task will return control to the MMI module. The Call Clearing task is composed of the Shutdown routine.

3.2.3 Gateway Driver

The Gateway Driver module processes the received messages from the NETBIOS, determines the required action, and calls the appropriate routine. Once initiated this module will run independently, regardless of the current operation being performed by the MMI module, until a Clear Calls request is entered by the operator. This module receives input messages from the NETBIOS and queues the message for processing by the appropriate Call Processing or Data Transfer modules. The following subsections describe the major tasks of the Gateway Driver module.

3.2.3.1 Incoming Packet Handling Task. This task processes the incoming packet from the NETBIOS by determining its type, command value, and process identifier. Process IDs are destination address translation table indices for listens and outstanding call list indices for other commands. Use of these identifiers allows the gateway to efficiently determine which previously established route to take. Protocol type, command and TCP process IDs are read from the packet. Other process identifiers are determined based on the protocol and command. X.25 listen process IDs are determined by searching through the translation table looking for the listen which has just completed. All other command process IDs are determined by searching the outstanding call table using the link identifier from the incoming packet as a key.

Once the required data has been found this task inserts the information into the Pending Action queue. The Pending Action queue is a FIFO queue used, along with the internal receive buffers, as the means to schedule a network activity for sequential processing. Use of this queue provides for orderly processing of the messages. Once this task has inserted the processing queue entry it transfers the relevant data to the buffers indicated by the determined process ID/command combination. After completion of this copy the task will transfer control to the Action Initiation task. If no incoming NETBIOS packet is detected and no queue entries exist that require processing this task will return control to the calling module.

The Incoming Packet Handling task is composed of the following routines:

- Chk_net - oversees the Gateway Driver processing
- Que_X25_LISTEN - processes received X25 listens
- Que_X25_Receive - receives X25 responses and queues the action
- Que_ISDN_act,Que_ISDN_control,Que_ISDN_receive - queues the various ISDN messages received
- Que_TCP_act - queues the TCP messages received.

3.2.3.2 Action Initiation Task. This task takes the queued actions from the Pending Action queue and initiates the appropriate Call Processing or Data Transfer module action. Received call requests and Hangup requests cause transfer of control to the Call Processing module. Received statistics packets cause control transfer to the MMI module. All other packets result in transfer to the Data Transfer module.

The action initiation task is composed of the following routine: Det_X25_act,Det_TCP_act,Det_ISDN_act - determines the action required for each individual protocol.

3.2.4 Call Processing

The Call Processing module processes incoming call and hangup requests performing any routing required and performing the physical setup or teardown on the destination link. This task also builds the outstanding call routing table used by the Data Transfer module to determine the destination of a data packet. The following subsections discuss the major tasks of this module.

3.2.4.1 Call Initiation Task. Upon receipt of an incoming call notification this task checks the list of currently outstanding calls for the incoming protocol type to determine if an empty queue slot is available. If an empty slot is found then routing of the message can begin and control is passed to the Routing task. If there are no empty queue slots then an error is indicated and the call request is rejected.

Once the Routing task has completed its determination of the next link to use this task issues a Call Request to the destination NETBIOS card. For TCP and X.25 calls this Call Request includes the source and destination address determined by the Routing task. For ISDN calls the Call Request is made to the destination node's local gateway address as indicated in the translation table. Receipt of a successful return from an ISDN call request will cause this task to format the destination type, destination address, source address, and if required TCP port numbers into an ISDN call header packet and transmit it to the remote gateway. Successful completion of the Call Request and, in the case of ISDN, the header transmit request will cause this task to regard the link as open, update the appropriate call level statistics, save the new link ID, request that data received on this link be passed to the gateway and return control to the calling module. Errors encountered in this processing will result in rejection of the requested call and clearing of the internal call list.

The Call Initiation task is composed of the following routines:

- Proc_X25_incall,Proc_TCP_incall,Proc_ISDN_incall - oversees the processing of the received call request
- Init_Dest_call - selects the output protocol type based on the stored output of the routing task

- Init_X25_call,Init_TCP_call,ISDN_d_link_open - performs the physical operations necessary to open a link to the destination

- Init_ISDN_call - oversees the processing required to open the ISDN link and transmit the call header

- Fmt_ISDN_hdr - formats and transmits the received ISDN call header packet

- ISDN_call_setup,X25_call_setup - sets up the transmission buffers for later use by the Data Transfer task.

3.2.4.2 Call Termination Task. Upon receipt of an incoming hangup request this task will hang up the requested call. For remote hangups this task will determine the process ID and link ID of the call by searching the call list or, in the case of TCP, from the process ID field of the request. For internal hangups (i.e., caused by some link error) this link ID is supplied by the calling module. Using the destination link ID information this task will initiate a hangup on the link. Hangups are performed by sending a Close or hangup command to the destination link's NETBIOS. For X.25 a Cancel is also issued to cancel any pending Receive Request. Hangup errors are displayed to the operator. Once the hangup has been completed this process will clear the internal call table entry to indicate that the connection no longer exists.

The call termination task is composed of the following routines:

- Hangup_dest - determines the destination type to hangup

- Hangup_ISDN_dest,Hangup_TCP_Dest - oversees the processing involved in closing these link types

- Disc_ISDN_link,X25_Hangup - formats and sends the Hangup Request to the NETBIOS

- Proc_TCP_Hangup,Proc_ISDN_Remcls - processes remote Hangup Requests received from the NETBIOS. X25 remote Hangups are received as part of the data and are processed in the received data routine.

3.2.4.3 Call Routing Task. This task processes incoming X.25 and TCP Call Requests and ISDN call header messages and determines the destination of the given set of addresses. The required information will be passed back to the Call Initiation task.

For X.25 and TCP calls, this task accepts a source and destination address and source link type. It then checks if the given addresses require translation into another address base. If translation is required the task changes the destination type and indicates that addresses corresponding to the new destination type should be used. This task will then check the list of outstanding calls for the destination address type and if the current call is a duplication then the it will be rejected.

For ISDN calls this task will wait until an ISDN call header packet has been received. Upon receipt of this packet the task will extract the destination type and look up the source and destination addresses in the translation table. Once this information has been determined the task will check for call duplication. Failure to find a translation table entry or detection of a duplicate call will result in call rejection.

The Call Routing task is composed of the following routines:

- Det_X25_dest,Det_TCP_dest - determines the destination of incoming TCP and X.25 calls

- Proc_ISDN_hdr - determines the destination of incoming ISDN calls.

3.2.4.4 Common Routines. The following common routines are used by the tasks of the Call Processing module:

- Conv_TCP_RX,Conv_ISDN_RX,Set_X25_addr - converts received addresses to internally usable formats

- Conv_TCP_Tx,Conv_X25_Tx - converts internal addresses to transmittable formats

- Perf_Com_NCB_Act - performs common actions required to format and ISDN command

- Send_Cmd_5c,Send_cmd_60 - performs the command send to the X25 and ISDN NETBIOS.

3.2.5 Data Transfer Processing

This module is responsible for performing the actions required to transfer data from the link it was received on to the destination link. It performs all error checking and data format translation. The following subsections discuss these tasks.

3.2.5.1 Received Data Task. This task processes data received on a link. Upon notification of received data this task determines which output link to transmit the received data on.

This determination is made by searching the outstanding call list for the current call entry for the received data link. The target link id from this entry, along with the location of the data and its size are then sent to the Transmit Data task. Upon completion of the transmission this task will send a new receive request to the source NETBIOS. Control will then be returned to the Gateway Driver module.

The Received Data task is composed of the following routine: Proc_Rx_TCP_Data,Proc_Rx_X25_data,Proc_Rx_ISDN_data - performs the above actions for each of the protocol types.

3.2.5.2 <u>Transmit Data Task</u>. This task accepts data from the Received Data task and transmits it on the selected link. Upon receipt of control from the Received Data task this task determines the target protocol type from the outstanding call table of the source protocol. This task then determines the location of the outstanding call entry in the target call list. The received data is then copied to the transmit buffer corresponding to this entry and a Transmit Request is sent to the target NETBIOS. Results of this request are analyzed and displayed to the operator.

The Transmit Data task is composed of the following routines:

- Init_tx_dest - determines the target link type
- Tx_TCP_msg,Tx_ISDN_msg,Tx_X25_msg - transmits the message on the target link.
- Proc_TCP_Tx_conc - analyses the results of TCP transmits.

3.3 NOTES ON CURRENT IMPLEMENTATION

The following limitations exist within the current gateway implementation.

- The gateway currently processes only those TCP messages destined for the TCP address specified at load time as the local address for the gateway. This is a limitation imposed by the TCP NETBIOS. Correction of this limitation would require customization of the NETBIOS.
- Most of the parameters for the gateway require that constants in the gateway software be changed and in some cases the program be recompiled to change their value. This limitation could be eliminated by the creation of an install option or a separate install program to set these parameters based on screen oriented input. This would require changes

to the gateway to remove the fixed size, static tables and replace them with dynamically sized allocated tables.

- Network Actions are performed as indivisible actions by the gateway driver. This means that there is idle time available for use while the NETBIOS processor is processing a requested action. This could be improved by dividing these actions into smaller tasks with boundaries at the point of NETBIOS access. This would require changes to the Gateway Driver that would increase its complexity and the implementation of dynamically sized and allocated buffers and activity queues for the NETBIOS interfaces.

- The method of accessing the translation table and the outstanding call lists depends heavily on sequential searches. This method results in a heavy speed penalty as table size and load increase. To improve this performance the access method should be changed to reduce or eliminate the searches on these tables. Preference should be given to those accesses involved in the data transfer processing as they appear to be the critical elements of gateway performance.

- The gateway does not currently support such protocol features as facilities handling for X.25 and user supplied open data.

4.0 MULTIPROTOCOL GATEWAY DEMONSTRATION

An operational demonstration was conducted on February 28, 1990, at the Booz, Allen & Hamilton Systems Resource Center (SRC), Protocol Development And Test Lab in Bethesda, Maryland. The purpose was to demonstrate the functionality of the multiprotocol gateway. The demonstration was designed to exercise the full functionality of the multiprotocol gateway. The equipment configuration and demonstration description are given below.

4.1 EQUIPMENT CONFIGURATION

The Protocol Development And Test Lab equipment configuration is shown in Exhibit 4-1. The configuration consisted of a DDN X.25 (TCP/IP) end user, a CCITT X.25 end user, the Multiprotocol Gateway, the TELEOS ISDN Central Office Emulator, and two ISDN end users. The ISDN end users were used to simulate remote X.25 and TCP/IP traffic as described below. The Multiprotocol Gateway is described in sections one through three of this document. The other components are described in more detail below.

The DDN X.25 (TCP/IP) end user consisted of a Compaq 386 PC with a Frontier Technologies Corporation AdCom2-I Intelligent Communication Controller card and Super DDN network software installed. The Adcom2-I interface was connected to the TCP/IP NIC interface in the Multiprotocol Gateway.

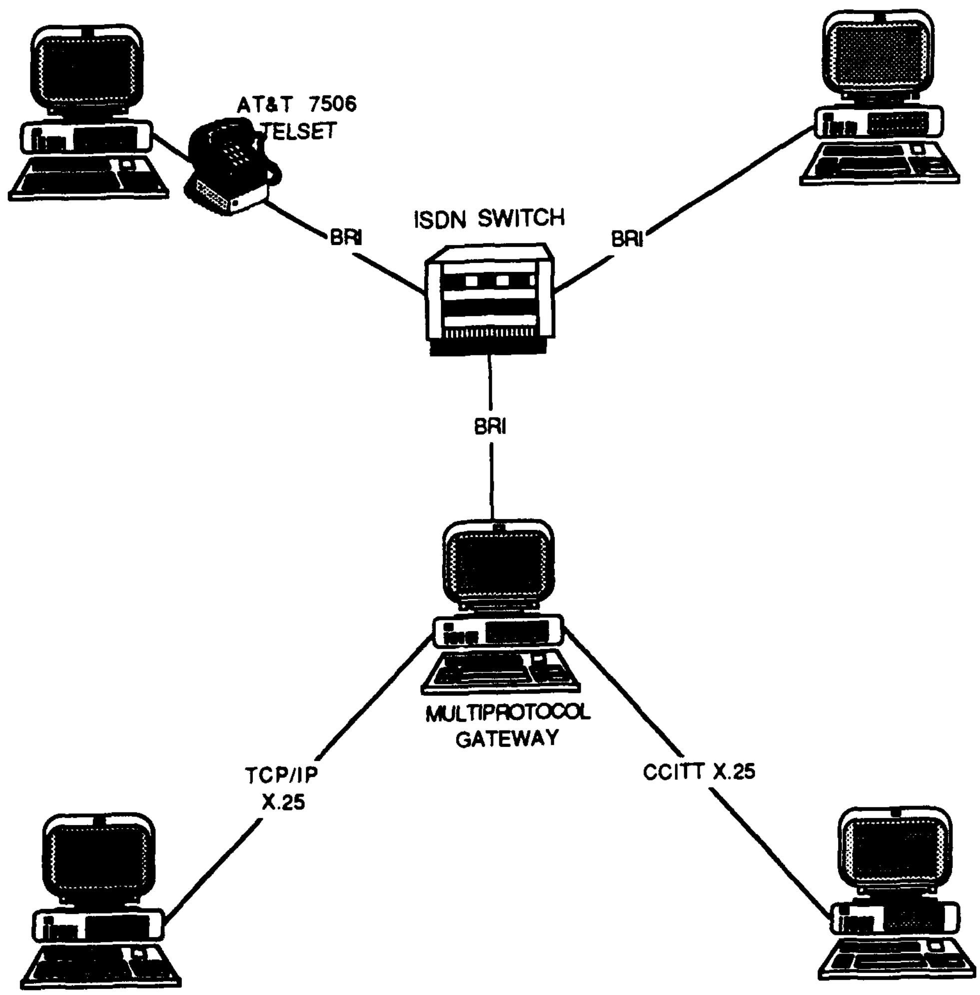
The CCITT X.25 end user consisted of a Wyse 286 PC with an EICON Technology Adapter card and associated software installed. The X.25 end user interface was connected to the CCITT X.25 NIC interface in the Multiprotocol Gateway.

The TELEOS ISDN Central Office Emulator is the TELEOS ASK200 Central Office Simulation System. The ASK200 is an ISDN switch that allows a developer to test and debug applications on ISDN terminal equipment. It provides the basic voice and data features of a 5ESS (4.1 Generic) ISDN switch. It also provides protocol trace capabilities for the Q.921 and Q.931 protocols on the D channel. This allows a developer to monitor in real time the protocol interactions between the ASK200 and any terminal equipment connected to it.

The first of the ISDN end users consisted of a Compaq 386 PC using an AT&T 7506 ISDN Telset as an ISDN terminal adapter. The 7506 ISDN Telset is an ISDN voice terminal with associated asynchronous data module which provides simultaneous voice and data capabilities. This ISDN end user was used to send/receive

EXHIBIT 4-1

Protocol Development Lab Equipment Configuration



interactive data transmissions and files to/from the DDN and X.25 end users. The other ISDN end user was composed of a Wyse 286 PC with a TELEOS B100PC terminal adapter board and software installed. Also running on this machine was a modified version of the gateway software that allowed the end user to accept multiple calls and store the incoming data. This ISDN end user was used to demonstrate multiple D channel call completion as described below.

4.2 OPERATIONAL DEMONSTRATION DESCRIPTION

To fully demonstrate the gateway in its intended environment an equipment configuration like the one shown in Exhibit 4-2 would be required. At the time of this demonstration, the equipment needed for the second gateway and additional end users were not available. However, the full functionality of the multiprotocol gateway was demonstrated using the equipment configuration shown in Exhibit 4-1 and by using a step-by-step approach. This approach is explained below.

During normal gateway operations, the gateway receives packets from either the ISDN, local X.25 end user, or local TCP/IP end user. When the multiprotocol gateway receives an incoming packet, it performs four basic functions:

- Determines the source of the packet (i.e., ISDN, X.25 end user, TCP/IP end user)
- Determines the destination of the packet (i.e., ISDN, X.25 end user, TCP/IP end user)
- Performs the appropriate protocol translation functions
- Sends packet to appropriate destination.

Incoming ISDN packets (to the gateway) originate from X.25 end users or TCP/IP end users that access the ISDN through a second gateway (refer to Exhibit 4-2). These end user packets have been translated from their native mode formats to ISDN format by the remote gateway. Therefore, remote end user traffic can be simulated by sending the appropriate ISDN packets to the local multiprotocol gateway. Local end user traffic can be simulated by supplying X.25 and TCP/IP packets to the gateway from the local X.25 and TCP/IP end users.

By exercising the above four basic functions, using the procedures described below, an operational multiprotocol gateway capability was demonstrated. The stepwise approach to demonstrating the multiprotocol gateway is summarized in

EXHIBIT 4-2

Operational Gateway Demonstration Environment

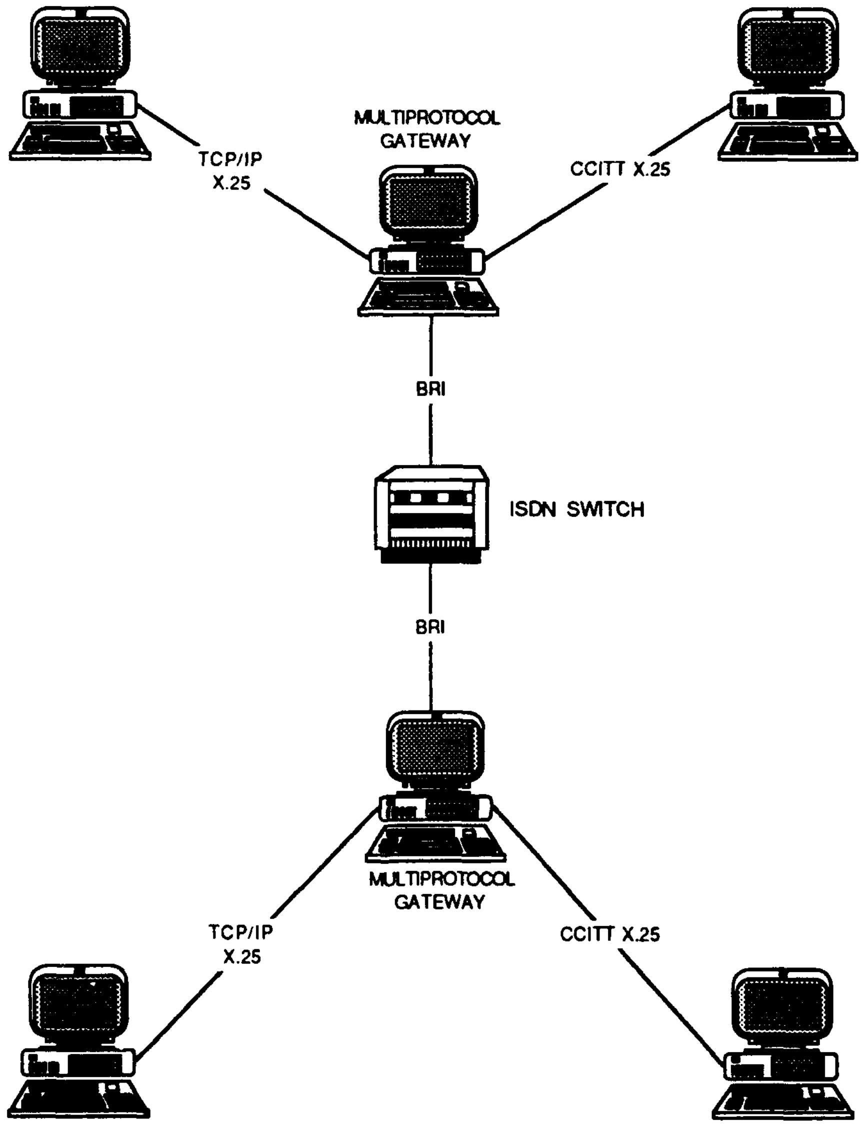


EXHIBIT 4-3

Demonstration Steps

Step 1: Local End User to Remote End User

X.25 ----- Gateway ----- ISDN Switch ----- ISDN End User

TCP/IP ----- Gateway ----- ISDN Switch ----- ISDN End User

Step 2: Remote End User to Local End User

ISDN End User ----- ISDN Switch ----- Gateway ----- X.25

ISDN End User ----- ISDN Switch ----- Gateway ----- TCP/IP

Step 3: Local X.25 End User to Local TCP/IP End User

X.25 ----- Gateway ----- TCP/IP

TCP/IP ----- Gateway ----- X.25

Step 4: Two Simultaneous Gateway Calls

X.25 ----- Gateway ----- ISDN Switch ----- ISDN End User
&
X.25 ----- Gateway ------ TCP/IP

X.25 ----- Gateway ----- ISDN Switch ----- ISDN End User
&
TCP/IP ----- Gateway ----- ISDN Switch ----- ISDN End User

Exhibit 4-3 and described in the paragraphs that follow.

The multiprotocol gateway functionality was demonstrated in four steps:

- Step 1 - local end user to remote end user communications
- Step 2 - remote end user to local end user communications
- Step 3 - local X.25 end user to local TCP/IP end user communications
- Step 4 - two simultaneous gateway calls.

As part of each step, both interactive data traffic and a standard text file (approximately 4855 bytes) were transferred between network end users through the multiprotocol gateway. After each gateway call, the gateway statistics were checked and compared to the actual data. The steps are described in more detail below.

In Step 1, local end user to remote end user communications through the gateway was demonstrated. First, an X.25 call was placed from the local (i.e., directly connected to the gateway) X.25 end user to an ISDN end user. The ISDN end user was used to simulate a remote X.25 or TCP/IP end user. This procedure was repeated with the local TCP/IP end user originating the call. Step 1 exercised gateway routing and protocol translation capabilities for outgoing (i.e., to the ISDN) calls.

Step 2 demonstrated remote end user to local end user communications through the gateway. A call was placed from the ISDN end user, through the ISDN switch, through the gateway, to the local X.25 end user. Next, this procedure was repeated using the local TCP/IP end user as the destination address for the call. This step showed gateway routing and protocol translation capabilities for incoming (i.e., from the ISDN) calls.

Completion of local calls through the gateway was demonstrated in Step 3. This was done by placing a call through the gateway from the local X.25 end user to the local TCP/IP end user. This procedure was repeated using the TCP/IP end user as the originator and the X.25 end user as the destination. This step exercised the gateway routing and protocol translation capabilities between X.25 and TCP/IP (transport layer and below) end users.

Two simultaneous gateway calls were established in Step 4. First, a gateway call was established between the local X.25 end user and an ISDN end user. This call was placed on hold at the X.25 end user machine and a second call was set up through the gateway from the X.25 end user to the local TCP/IP end user. Next, a call was established through the gateway between the local X.25 end user and a remote ISDN end user. During this call, a second call was established through the gateway between the local TCP/IP end user and the ISDN end user. Step 4 showed the gateway's ability to successfully handle more than one call at a time.

5.0 RESULTS AND CONCLUSIONS

A multiprotocol gateway has been developed to support network reconstitution in the event of network disruption. The gateway allows CCITT X.25 users as well as DDN TCP/IP X.25 users to use the ISDN D channel packet facilities as a transport mechanism to reconstitute disrupted portions of their networks. The gateway allows CCITT X.25 end users and DDN TCP/IP X.25 end users to communicate with each other by providing a protocol translation capability between CCITT X.25 packets, DDN TCP/IP X.25 packets, and ISDN D channel packets (transport level and below) in a full-duplex mode of operation. The gateway was implemented on a Compaq 386 platform and uses TELEOS ISDN, EICON X.25, and Frontier Technologies Corporation DDN X.25 protocol processing hardware and software drivers. Gateway software, which interfaces with the protocol processors and software drivers, was developed in the "C" language. The gateway software also performs address translation and statistics gathering. Both link statistics and gateway environment statistics (e.g., packets processed, data transmitted, data received) can be collected and displayed.

The multiprotocol gateway demonstration, which took place at the Booz, Allen & Hamilton Systems Resource Center in Bethesda, Maryland, showed that the gateway is capable of providing the services described above. Gateway functionality was demonstrated in a four-step procedure. This is discussed in Section 4. Results of the demonstration are discussed below.

In Step 1, locally originated calls were established from an X.25 end user and a TCP/IP end user, through the gateway, to a remote X.25 and TCP/IP end user (simulated by an ISDN end user) respectively. Step 2 demonstrated that the gateway can support calls originating from remote end users with local X.25 and TCP/IP destinations. A local X.25 to local TCP/IP end user call was established in Step 3. In each of these steps, interactive traffic was sent across each connection. Also, standard text files ranging in size from 4900 bytes to 21,500 bytes were sent across the connections. In every case, the gateway performed as expected. No problems were encountered.

Step 4 showed that the multiprotocol gateway can support multiple, simultaneous calls. In addition to the specified procedure, 3 simultaneous calls were established through the gateway. A call was established between the local X.25 end user and a remote ISDN end user (Note: the ISDN end users were used to simulate remote X.25 and TCP/IP end users). Next, a call was established between the local TCP/IP end user and a remote ISDN end user. At this point, a third call was established between the local X.25 end user and the local TCP/IP end user. Standard text files were sent between the X.25 and ISDN end users, and the TCP/IP and ISDN end users simultaneously. Although transmission of the files was a little slower than that experienced in Steps 1 through 3, the files were transmitted without any problems.

At the end of each step, gateway statistics were checked using the menu based Man-Machine Interface. The gateway's MMI was clear and easy to use. Using the gateway statistics utilities during data transmission slowed gateway throughput considerably. This was expected. Gateway timing parameters were also measured using utilities contained in the software. Average delay for call setup - for the equipment configuration in the lab - was on the order of 160 milliseconds. Call setup delay will depend on components in addition to the gateway such as ISDN switch and destination end user. Average data handling delay was approximately 79 milliseconds. Average queuing delay was less than 15 milliseconds. This is well within the specified gateway timing parameters.

It should be noted that the multiprotocol gateway software is the result of a proof-of-concept gateway development. There are several features that can be incorporated into the gateway to make the gateway more functional. These are described in Section 3.3. From a NCS perspective, these features should be researched and implemented because they provide insight and guidance on how these features should be provided for in a production version of the gateway.

In conclusion, the gateway performed as expected. It was able to automatically identify the protocol of the packet being received and switch to the appropriate transformation and mapping routines. The demonstration exercised the gateway's routing and protocol/address translation capabilities and demonstrated the gateway's ability to support multiple simultaneous calls.

L I S T O F A C R O N Y M S

LIST OF ACRONYMS

ACRONYM	DESCRIPTION
BRI	Basic Rate Interface
bps	Bits per second
CCITT	The International Telegraph and Telephone Consultative Committee
DCE	Data circuit-terminating equipment
DPP	Data Protocol Processing
DTE	Data terminal equipment
DDN	Defense Data Network
DOD	Department of Defense
FIFO	First In First Out
GTM	Gateway Translation Matrix
ID	Identifier
ISDN	Integrated Services Digital Network
IP	Internet Protocol
kbps	Kilobits per second
kB	Kilobytes
MB	Megabytes
MPU	Master Processing Unit
MMI	Man-Machine Interface
NS/EP	National Security and Emergency Preparedness
NIC	Network Interface Card

OMNCS	Office Of The Manager, National Communications System
PC	Personal Computer
RAM	Random Access Memory
SDL	Specification and Description Language
SIP	Signal Input Processor
TA	Terminal Adaptor
TCP	Transmission Control Protocol